Contract No. H2020 – 730842

SEMANTIC TRANSFORMATIONS FOR RAIL TRANSPORTATION

D5.3 – Demonstration platform ready for test

Due date of deliverable: 30/04/2018

Actual submission date: 11/09/18

Leader/Responsible of this Deliverable: OLTIS Group

Reviewed: Yes

Document status		
Revision	Date	Description
1	20.06.2018	First draft
2	24.07.2018	Final version
3	07.08.2018	Final version after review
4	11/09/2018	Quality Check

Project funded from the European Union's Horizon 2020 research and innovation programme		
Dissemination Level		
PU	Public	X
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

Start date of project: 01/11/2016 Duration: 24 months

EXECUTIVE SUMMARY

This document states the readiness of the selected demonstration platform to perform the tests of the ST4RT Converter. It comprises the description of the test scenario and indicates the steps of such scenario that have already been successfully tested. This document also specifies the various inputs, e.g. annotations or SPARQL queries, which are necessary for successful conversion.

ABBREVIATIONS AND ACRONYMS

Abbreviation	Description
FSM	Full Service Model
GUI	Graphical User Interface
RDF	Resource Description Framework
REST	REpresentational State Transfer
RSP	Rail Service Provider
RU	Railway Undertaking
SOAP	Simple Object Access Protocol
SPARQL	Simple Protocol And RDF Query Language
ST4RT	Semantic Transformations for Rail Transportation
VPN	Virtual Private Network
XML	Extensible Markup Language
XSD	XML Schema Definition

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The aim of the document is to briefly describe the testing platform inside the HEROS platform and to introduce the ST4RT converter, as well as both FSM and 918 annotations. In the next part, the test scenario is described in the form of conversion of messages between SNCF and Trenitalia, i.e. the mutual conversion of messages in FSM and 918 formats. In connection with the test scenario, the test messages (FSM PreBookRequest = 918 ReservationRequest and vice versa, i.e. 918 ReservationReply message = FSM PreBookResponse) are specified. Last but not least, this document describes the individual steps of the test scenario that have been successfully tested, including the conversion itself.

2. TESTING PLATFORM

The test environment is located within the HEROS platform provided by Hit Rail. It is accessed through the HERMES VPN, also provided by Hit Rail. An access to the network must be allowed.

The HEROS platform is based around a Middleware, or Broker, called NIKLAS. According to the originally intended architecture, the ST4RT Converter was supposed to be placed inside the Broker – see Figure 1.

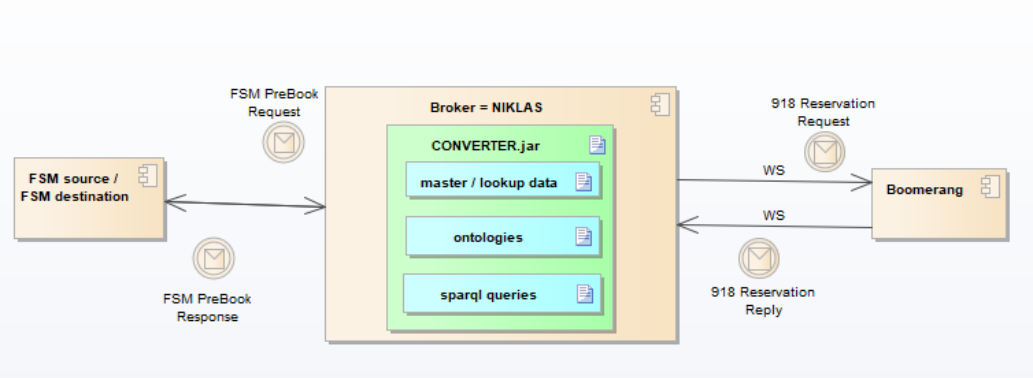

Figure 1 – Originally intended architecture of Converter deployment

After deployment of the ST4RT Converter inside the Broker, it turned out that it didn't work as planned. The ST4RT Converter cannot be directly inserted into the Broker, because the Broker requires older versions of some libraries than the ST4RT Converter. However, it is not possible to use two different versions of the same library concurrently in the Broker, therefore the architecture had to be changed.

The new architecture is illustrated in the following diagram – see Figure 2.

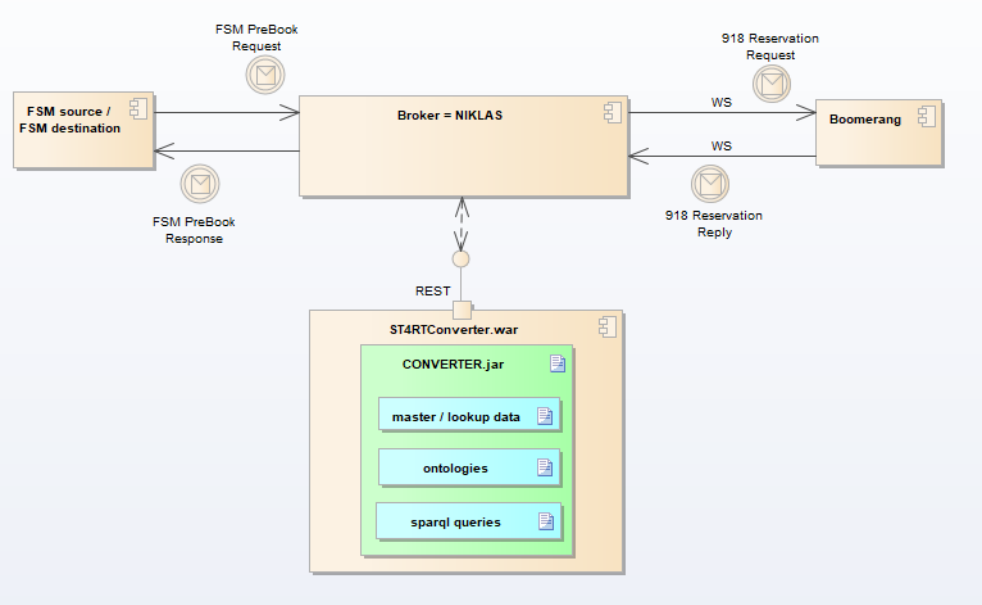

Figure 2 – New architecture of Converter deployment

The ST4RT Converter is embedded in the standalone small application without its own GUI. It communicates with the Broker through its REST interface. The conversion proceeds as follows:

1. Broker calls Converter's method, forwards the message string and waits for return value.
2. Converter converts an incoming string into the outgoing string.
3. Converter returns the converted message to Broker.

2.1 NIKLAS BROKER

To be able to send the FSM PreBookRequest to the Broker, it was necessary to implement a web service where the FSM message may be sent to.

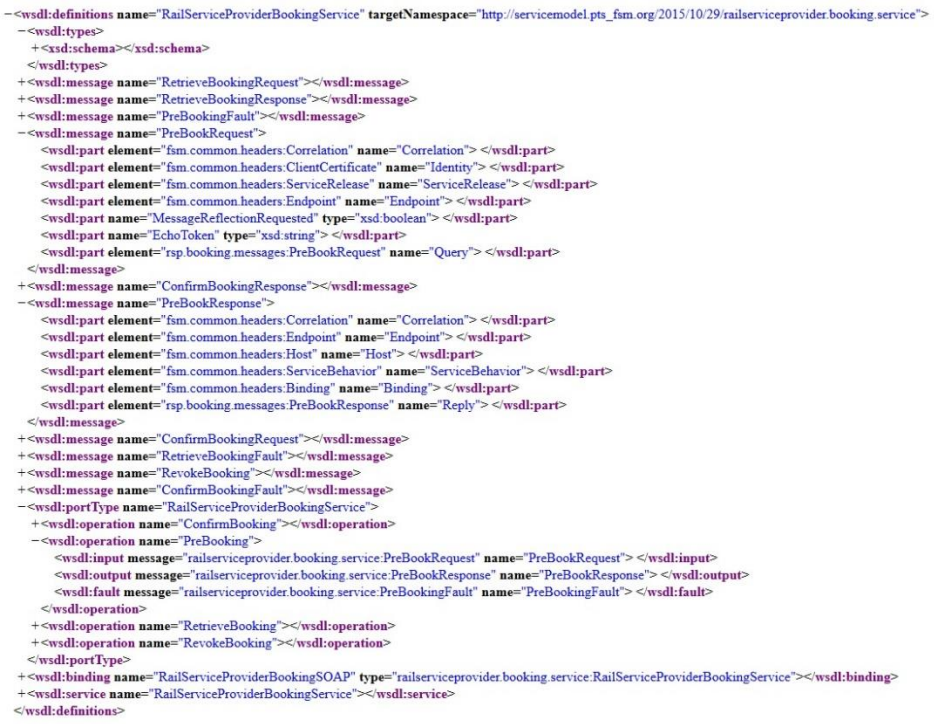

Figure 3 – Example of implemented FSM WSDL provided by Broker

The next step was to set up the routing configuration between „users" who use messages in different formats. The configuration was set up according to the test scenario, i.e. SNCF sends and receives FSM messages and Trenitalia sends and receives 918 messages. Both users use a Web service communication channel and the expected file format is XML.

2.2 BOOMERANG

The Boomerang application is a testing tool which can receive 918 request and generate corresponding 918 reply for BerthRequest message type.

2.3 SOAPUI

SoapUI is the world's leading Functional Testing tool for SOAP and REST testing. With its easy-to-use graphical interface and enterprise-class features, SoapUI enables to design, build and perform automated functional, regression and load tests easily and quickly. In a single test environment, SoapUI provides the complete test coverage – from SOAP and REST-based Web services, to JMS enterprise messaging layers, databases, Rich Internet Applications, and much more. [1]

The SoapUI tool is used for sending FSM message and for displaying the corresponding FSM response.

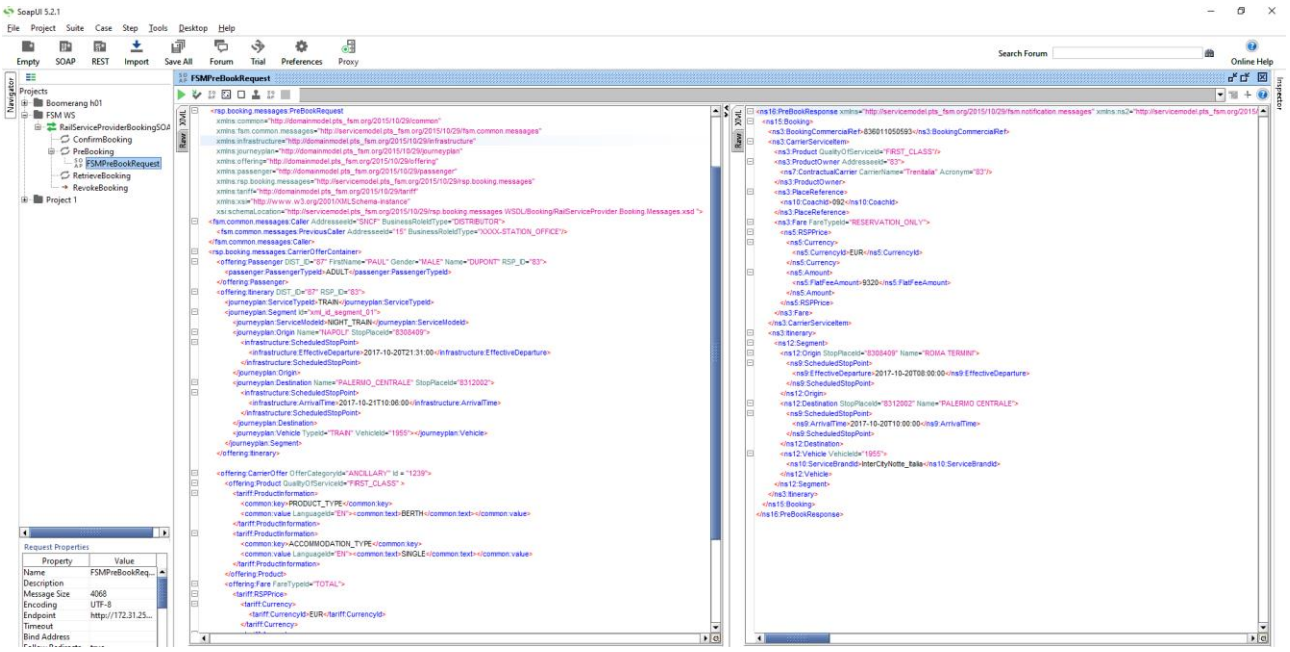

Figure 4 – Sent FSM request, received FSM response through SoapUI tool

3. ST4RT CONVERTER

This chapter specifies the building blocks required for a successful conversion from FSM message to 918 format and vice versa. The successful conversion has to create valid messages which are passed to the Broker that sends them to the destination system. The valid message which is sent by the Broker proves that the testing platform works properly and is ready to use. In case of synchronous communication (this situation), the Broker has to send both request and corresponding response.

3.1 FSM AND 918 ANNOTATIONS

FSM PreBookRequest and FSM PreBookResponse were annotated so that they are in line with 918 ReservationRequest and 918 ReservationReply in terms of their mandatory elements. The FSM messages contain elements that are not so strictly required – almost all these elements are optional in XSD.

Mandatory elements of 918 ReservationRequest and 918 ReservationReply were annotated according to the ontology items. Property names are defined in the ontology and agreed to be used for the defined concept.

AllocatedTrainType.java

```
66          @Link(propertyName = "mobility:hasTravelEpisode", nodeType = Link.Nod
67          @Link(propertyName = "st4rt:hasOrigin", nodeType = Link.NodeType.Shar
68          propertyName = "infrastructure:hasStopPlaceName")
69      protected String departureStationName;
70      @XmlElement(name = "ArrivalStationName", required = true)
71      @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
72      @XmlSchemaType(name = "token")
73      @RdfProperty(links = {
74          @Link(propertyName = "travel:hasItineraryOffer", nodeType = Link.Node
75          @Link(propertyName = "st4rt:hasItinerary", nodeType = Link.NodeType.S
76          @Link(propertyName = "mobility:hasTravelEpisode", nodeType = Link.Nod
77          @Link(propertyName = "st4rt:hasDestination", nodeType = Link.NodeType
78          propertyName = "infrastructure:hasStopPlaceName")
79      protected String arrivalStationName;
80      @XmlElement(name = "ServiceBrand", required = true)
81      protected ServiceBrandInformationType serviceBrand;
82      @XmlElement(name = "Category")
83      protected TrainCategory918Type category;
84      @XmlElement(name = "Train", required = true)
85      @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
86      @XmlSchemaType(name = "token")
87      @RdfProperty(links = {
88          @Link(propertyName = "travel:hasItineraryOffer", nodeType = Link.Node
89          @Link(propertyName = "st4rt:hasItinerary", nodeType = Link.NodeType.S
90          @Link(propertyName = "mobility:hasTravelEpisode", nodeType = Link.Nod
91          @Link(propertyName = "travel:isForTransportationService", nodeType =
92          @Link(propertyName = "transport:hasEquipment", nodeType = Link.NodeTy
93          propertyName = "transport:hasTransportationServiceCode")
94      protected String train;
```

Vehicle.java

```
58
59      public void setRdfId(RdfKey theId) {
60          mIdSupport.setRdfId(theId);
61      }
62
63      @XmlElement(name = "ServiceBrandId")
64      @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
65      @XmlSchemaType(name = "NMTOKEN")
66      @RdfProperty(links = {
67              @Link(propertyName = "transport:hasServiceBrandCode", nodeType
68              propertyName = "st4rt:hasCVValue")
69      @Query(name = "FSM_LW_ServiceBrand", type = Query.QueryType.Lowering, (
70      protected List<String> serviceBrandId;
71      @XmlElement(name = "OnboardServiceCategoryId")
72      @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
73      @XmlSchemaType(name = "NMTOKEN")
74      protected List<String> onboardServiceCategoryId;
75      @XmlAttribute(name = "ExternalId")
76      @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
77      @XmlSchemaType(name = "NMTOKEN")
78      protected String externalId;
79      @XmlAttribute(name = "TypeId")
80      @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
81      protected String typeId;
82      @XmlAttribute(name = "VehicleId")
83      @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
84      @XmlSchemaType(name = "NMTOKEN")
85      @RdfProperty(propertyName = "transport:hasTransportationServiceCode")
86      protected String vehicleId;
```

Figure 5 – Mandatory Train (number) in 918 mapped to optional VehicleId in FSM

Figure 5 shows an example of mapping between mandatory 918 element and optional FSM element with the same meaning (green lines).

3.2 SPECIFIC SPARQL QUERIES

The file *sparql_queries.xml* contains queries needed to reshape the RDF graph structure from a general form to a form necessary for 918 or FSM standards.

Some of the queries are used for filling in the default values. The default values of some 918 / FSM elements or attributes are assigned by a SPARQL query, because there are no equivalent values in the source message.

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "DialogueType", propOrder = {
    "dispositionElement",
    "requestDate"
})
@Queries({
        @Query(name = "918_LW_Default_Dialog", outputs = {"test", "applicationVersion", "requestDate"}, type = Query.QueryType.Lowering)})
public class DialogueType implements SupportsRdfId {

    @XmlTransient
    private SupportsRdfId mIdSupport = new SupportsRdfIdImpl();

    public RdfKey getRdfId() {
        return mIdSupport.getRdfId();
    }

    public void setRdfId(RdfKey theId) {
        mIdSupport.setRdfId(theId);
    }

    @XmlElement(name = "DispositionElement")
    protected String dispositionElement;
    @XmlElement(name = "RequestDate", required = true)
    @XmlSchemaType(name = "date")
```

Figure 6 – Calling a SPARQL query to fill the default values in Dialog attributes

```
<query>
    <name>918_LW_Default_Dialog</name>
    <outputs>
        <output>
            test
        </output>
        <output>
            applicationVersion
        </output>
        <output>
            requestDate
        </output>
    </outputs>
    <sparqlQuery><![CDATA[
         SELECT ?applicationVersion ?test ?requestDate
         WHERE
         {
         BIND ("true" AS ?test).
         BIND (0 AS ?applicationVersion).
         BIND(NOW() AS ?requestDate).
         }]]>
     </sparqlQuery>
</query>
```

Figure 7 – Example of a specific SPARQL query with output values

3.3 MASTER / LOOK UP DATA

Master / lookup data is a part of the ontology files. It contains so called Individuals which are instances of an object.

The value **SINGLE** in FSM means only one person in the compartment with the Berth. However, there are two possible values in 918 for the compartment for one person:

1. The value **L** in 918 ReservationRequest in the attribute *Type* in *Berth* element.
2. The value **1** in 918 ReservationReply in the attribute *Code* in *Place* element.

The first one is necessary to create a valid 918 ReservationRequest from FSM PreBookRequest and the second one is necessary to create the corresponding FSM PreBookResponse from related 918 ReservationReply, although the element *ProductInformation* (with the following content) is optional:

```
<tariff:ProductInformation>
        <common:key>ACCOMMODATION_TYPE</common:key>
        <common:value LanguageId="EN">
                <common:text>SINGLE</common:text>
        </common:value>
</tariff:ProductInformation>
```

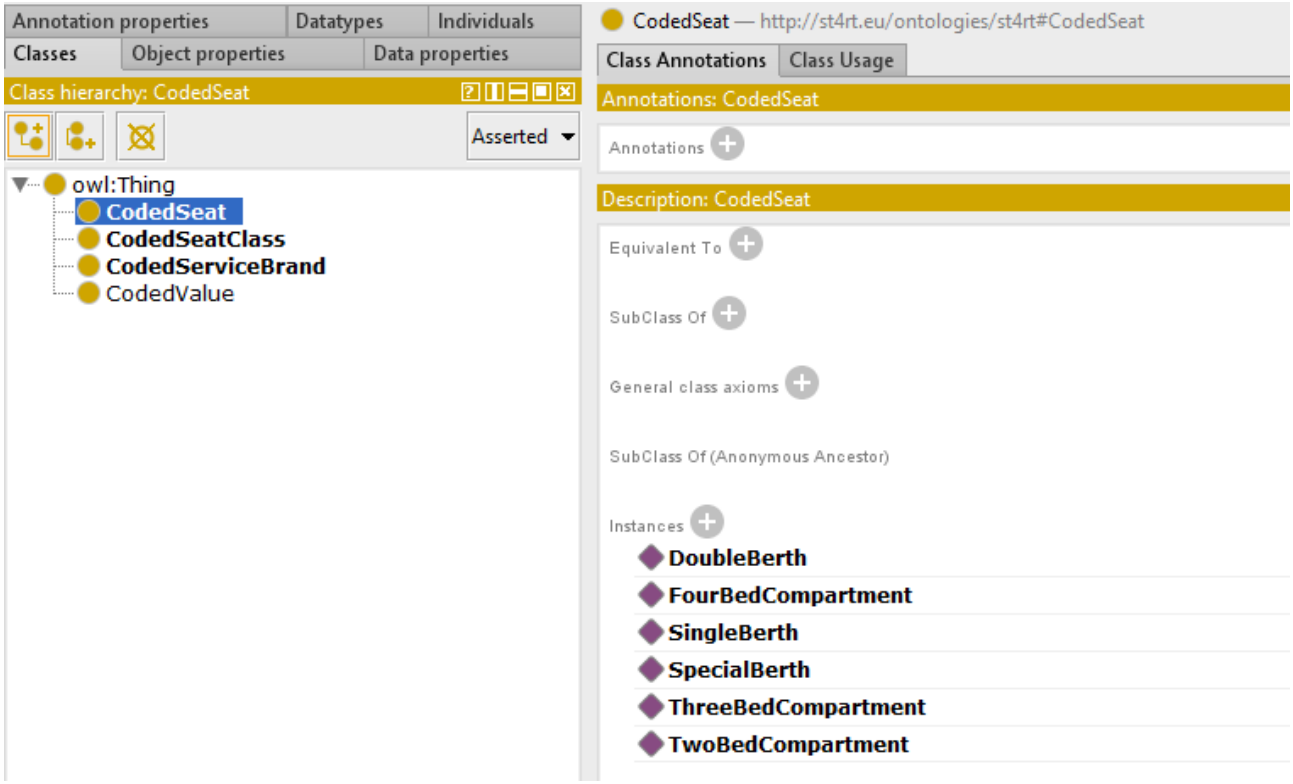

Figure 8 – Sample of accommodation types from Ontology

4. "DEMONSTRATION PLATFORM READY" VERIFICATION

This chapter states what has been done to verify that the demonstration platform is ready. The verification is based on the test scenario which is specified in the deliverable D5.1 and which describes in detail the communication loop. To verify the readiness of the platform, the last step of the loop, when the Broker returns the response in the synchronous communication, must be confirmed. The annotations and related SPARQL queries for mandatory elements had to be ready in order to create corresponding elements of the message. Created messages have been validated against XSD, as web services cannot receive invalid messages. The main test scenario from the D5.1 deliverable, whose steps were used to verify the synchronous communication loop readiness, is as follows:

We imagine that Mr Paul Dupont, an adult French citizen, is going to fly to Rome and then wants to go to Palermo by the Intercity night train 1955 of Trenitalia (service brand code 96), leaving Roma Termini (location code 08409) on 20th October at 21:31 and reaching Palermo Centrale (location code 12002) on 21st at 10:06.

In a previous step, he received the carrier offer detailing the services offered on that vehicle and the costs. He already has a transport title (e.g. an Interrail pass) and only needs the reservation. On 27th

September 2017 he goes to a counter of SNCF's Gare du Nord (location code 01519) to buy the reservation. He wants to travel in a single compartment first class. The terminal serving the customer has terminal number 15.

We imagine SNCF's Reservation system (code 87) is using FSM and therefore issues an FSM Booking_PreBookRequest.

We imagine Trenitalia's reservation system (code 83) is using 918 and therefore the converter translates the request into a 918 uic_reservationbookrq.

The request is sent to Hit Rail's Boomerang tool, that simulates Trenitalia's reservation system and issues a coherent 918 uic_reservationbookrp with reference number 836011050593, for a berth in coach 092, place 021.

The converter translates it into an FSM Booking_PreBookResponse. [2]

In brief:

- FSM request is converted to 918 (Berth) request and corresponding 918 reply is converted to FSM response.
- SNCF uses the FSM standard, therefore SNCF is represented by the SoapUI which is capable of sending FSM messages.
- Trenitalia uses the 918 standard, therefore this RU is represented by the Boomerang tool.

4.1 TEST MESSAGE

In order to prove that the demonstration platform is ready, the following **starting test message (FSM PreBookRequest)** was taken from D5.1 and updated to become valid.

SNCF (SoapUI tool) sends the following message with the SOAP Envelope to Trenitalia (Boomerang) via Broker. The SOAP Header is filled in only with mandatory items because there is no logic in processing the message by the SoapUI tool.

FSM PreBookRequest

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:fsm="http://servicemodel.pts_fsm.org/2015/10/29/fsm.common.header">
   <soap:Header>
      <fsm:ClientCertificate />
      <fsm:Correlation Message.GUID="03abcABC-2345-45fF-67eE-890abcdABCD8"/>
   </soap:Header>
   <soap:Body>

      <rsp.booking.messages:PreBookRequest
      xmlns:actors="http://domainmodel.pts_fsm.org/2015/10/29/actors"
      xmlns:booking="http://domainmodel.pts_fsm.org/2015/10/29/booking"
      xmlns:common="http://domainmodel.pts_fsm.org/2015/10/29/common"
      xmlns:connection="http://domainmodel.pts_fsm.org/2015/10/29/connection"
      xmlns:fsm.common.messages="http://servicemodel.pts_fsm.org/2015/10/29/fsm.common.messages"
      xmlns:fsm.notification.messages="http://servicemodel.pts_fsm.org/2015/10/29/fsm.notification
      .messages"
      xmlns:fulfilment="http://domainmodel.pts_fsm.org/2015/10/29/fulfilment"
      xmlns:infrastructure="http://domainmodel.pts_fsm.org/2015/10/29/infrastructure"
```

```
xmlns:journeyplan="http://domainmodel.pts_fsm.org/2015/10/29/journeyplan"
xmlns:offering="http://domainmodel.pts_fsm.org/2015/10/29/offering"
xmlns:passenger="http://domainmodel.pts_fsm.org/2015/10/29/passenger"
xmlns:payment="http://domainmodel.pts_fsm.org/2015/10/29/payment"
xmlns:roles="http://domainmodel.pts_fsm.org/2015/10/29/roles"
xmlns:rsp.booking.messages="http://servicemodel.pts_fsm.org/2015/10/29/rsp.booking.messages"
xmlns:tariff="http://domainmodel.pts_fsm.org/2015/10/29/tariff"
xmlns:transportation="http://domainmodel.pts_fsm.org/2015/10/29/transportation"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://servicemodel.pts_fsm.org/2015/10/29/rsp.booking.messages
WSDL/Booking/RailServiceProvider.Booking.Messages.xsd ">

  <fsm.common.messages:Caller AddresseeId="SNCF" BusinessRoleIdType="DISTRIBUTOR">
    <fsm.common.messages:PreviousCaller AddresseeId="15" BusinessRoleIdType="XXXX-
STATION_OFFICE"/>
  </fsm.common.messages:Caller>

  <rsp.booking.messages:CarrierOfferContainer>

    <offering:Passenger DIST_ID="87" FirstName="PAUL" Gender="MALE" Name="DUPONT"
RSP_ID="83">
      <passenger:PassengerTypeId>ADULT</passenger:PassengerTypeId>
    </offering:Passenger>

    <offering:Itinerary DIST_ID="87" RSP_ID="83">
      <journeyplan:ServiceTypeId>TRAIN</journeyplan:ServiceTypeId>
      <journeyplan:Segment Id="xml_id_segment_01">
        <journeyplan:ServiceModeId>NIGHT_TRAIN</journeyplan:ServiceModeId>
        <journeyplan:Origin Name="ROMA_TERMINI" StopPlaceId="8308409">
          <infrastructure:ScheduledStopPoint>
            <infrastructure:EffectiveDeparture>2017-10-
20T21:31:00</infrastructure:EffectiveDeparture>
          </infrastructure:ScheduledStopPoint>
        </journeyplan:Origin>
        <journeyplan:Destination Name="PALERMO_CENTRALE" StopPlaceId="8312002">
          <infrastructure:ScheduledStopPoint>
            <infrastructure:ArrivalTime>2017-10-21T10:06:00</infrastructure:ArrivalTime>
          </infrastructure:ScheduledStopPoint>
        </journeyplan:Destination>
        <journeyplan:Vehicle TypeId="TRAIN" VehicleId="1955"></journeyplan:Vehicle>
      </journeyplan:Segment>
    </offering:Itinerary>

    <offering:CarrierOffer OfferCategoryId="ANCILLARY">
      <offering:Product QualityOfServiceId="FIRST_CLASS" >
              <tariff:ProductInformation>
                        <common:key>PRODUCT_TYPE</common:key>
                           <common:value
LanguageId="EN"><common:text>BERTH</common:text></common:value>
                  </tariff:ProductInformation>
                  <tariff:ProductInformation>
                        <common:key>ACCOMMODATION_TYPE</common:key>
                           <common:value
LanguageId="EN"><common:text>SINGLE</common:text></common:value>
                  </tariff:ProductInformation>
      </offering:Product>
      <offering:Fare FareTypeId="TOTAL">
                  <tariff:RSPPrice>
                        <tariff:Currency>
                            <tariff:CurrencyId>EUR</tariff:CurrencyId>
                        </tariff:Currency>
                        <tariff:Amount>
                            <tariff:FlatFeeAmount>7000</tariff:FlatFeeAmount>
                        </tariff:Amount>
            </tariff:RSPPrice>
      </offering:Fare>
    </offering:CarrierOffer>

  </rsp.booking.messages:CarrierOfferContainer>
</rsp.booking.messages:PreBookRequest>

  </soap:Body>
</soap:Envelope>
```

The FSM PreBookRequest message is converted to the following **918 ReservationRequest** message by the ST4RT Converter.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:ReservationRequest
xmlns:ns2="http://www.uic-asso.fr/xml/passenger/02"
xmlns:ns3="http://www.uic-asso.fr/xml/passenger/reservation/01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.uic-asso.fr/xml/passenger/reservation/01
xml_schema_files/uic_reservationbookrq.xsd">

<Requestor RequestingSystem="83" SendingSystem="87">
        <Terminal Type="0">
                <Number>15</Number>
        </Terminal>
</Requestor>

<Dialogue ApplicationVersion="0" DialogueId="1234" Test="false">
        <RequestDate>2017-09-27</RequestDate>
</Dialogue>

<BerthRequest>
        <RequestedTrain SequenceId="01">
                <Train>1955</Train>
                <DepartureDate>2017-10-20</DepartureDate>
                <DepartureStation>
                        <ns2:Country>83</ns2:Country>
                        <ns2:LocalCode>08409</ns2:LocalCode>
                </DepartureStation>
                <ArrivalStation>
                        <ns2:Country>83</ns2:Country>
                        <ns2:LocalCode>12002</ns2:LocalCode>
                </ArrivalStation>
        </RequestedTrain>
        <Passengers>1</Passengers>
        <Berths Type="L">1</Berths>
</BerthRequest>

</ns3:ReservationRequest>
```

Values of unused FSM elements are lost (for example Passenger's Names, Fares, etc.).

The default values of some 918 elements or attributes are assigned by a SPARQL query, because there are no equivalent values in the FSM message:

- Type = 0
- ApplicationVersion = 0
- DialogueId = 1234
- Test = false
- SequenceId = 01

The attribute *Context* in the elements *DepartureStation* and *ArrivalStation* is not necessary. If nothing is set up in the message, then the default value of *918* is used.

Trenitalia (Boomerang) receives the above mentioned 918 ReservationRequest and replies with 918 ReservationReply. The 918 ReservationReply contains corresponding values that were gathered from the 918 ReservationRequest.

The **918 ReservationReply** is always positive, the Boomerang tool does not generate an answer message with the element NegativeReply.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:ReservationReply
xmlns:ns2="http://soapheader.hitrail.com/"
xmlns:ns3="http://www.uic-asso.fr/xml/passenger/02"
xmlns:ns4="http://www.uic-asso.fr/xml/passenger/reservation/01"
InformationOnly="true">
            <Requestor RequestingSystem="87" SendingSystem="83">
                <Terminal Type="0">
                    <Number>15</Number>
                </Terminal>
            </Requestor>
            <Dialogue ApplicationVersion="0" DialogueId="1234" Test="true">
                <RequestDate>2018-06-28+02:00</RequestDate>
            </Dialogue>
            <Allocation SequenceId="1">
                <Reference AllocatingSystem="83" LocalReference="6011050593"/>
                <BerthAllocation>
                    <AllocatedTrain>
                        <DepartureStationName>ROMA TERMINI</DepartureStationName>
                        <ArrivalStationName>PALERMO CENTRALE</ArrivalStationName>
                        <ServiceBrand>
                            <ns3:Code>96</ns3:Code>
                        </ServiceBrand>
                        <Train>1955</Train>
                        <DepartureDateTime>2017-10-20T08:00:00</DepartureDateTime>
                        <ArrivalDateTime>2017-10-20T10:00:00</ArrivalDateTime>
                    </AllocatedTrain>
                    <Passengers>1</Passengers>
                    <CoachNumber>092</CoachNumber>
                    <AllocatedPlaces>
                        <Place Code="1" Place="028"/>
                    </AllocatedPlaces>
                    <BookedOffer>
                        <Price>
                            <Amount>5743</Amount>
                            <PartialPrice>
                                <Passengers>0</Passengers>
                                <Amount>5743</Amount>
                            </PartialPrice>
                        </Price>
                    </BookedOffer>
                </BerthAllocation>
                <Carrier>83</Carrier>
            </Allocation>
</ns4:ReservationReply>
```

The Boomerang swaps the values in elements *RequestingSystem* and *SendingSystem* to keep the information, who is the author of the message and where it should be sent to.

The Boomerang sets up some default values and some generated values for some elements or attributes that simulate RU decisions or Place availability.

Default values:

- SequenceId = 01
- LocalReference = 6011050593
- Code = 96
- DepartureDateTime = DepartureDate at 21:30:00
- ArrivalDateTime = DepartureDateTime + *12 h 40 min*
- CoachNumber = 092

Generated values:

- RequestDate
- Place
- Amount *[currency is always set up to EUR]*

The values of elements *DepartureStationName* and *ArrivalStationName* are taken from Boomerang's built-in Location code list. The XSD defines station names for 918 reply only, what causes troubles in the next conversion.

Berths Type value **L** (used in 918) is assigned to value **1** in the attribute *Code* in the *Place* element with the help of Boomerang logic.

This 918 ReservationReply message is converted to the following **FSM PreBookResponse**.

```
<ns16:PreBookResponse
xmlns="http://servicemodel.pts fsm.org/2015/10/29/fsm.notification.messages"
xmlns:ns2="http://servicemodel.pts fsm.org/2015/10/29/fsm.common.messages"
xmlns:ns4="http://domainmodel.pts fsm.org/2015/10/29/common"
xmlns:ns3="http://domainmodel.pts_fsm.org/2015/10/29/booking"
xmlns:ns6="http://domainmodel.pts_fsm.org/2015/10/29/offering"
xmlns:ns5="http://domainmodel.pts_fsm.org/2015/10/29/tariff"
xmlns:ns8="http://domainmodel.pts fsm.org/2015/10/29/payment"
xmlns:ns7="http://domainmodel.pts fsm.org/2015/10/29/roles"
xmlns:ns13="http://domainmodel.pts fsm.org/2015/10/29/connection"
xmlns:ns9="http://domainmodel.pts_fsm.org/2015/10/29/infrastructure"
xmlns:ns12="http://domainmodel.pts fsm.org/2015/10/29/journeyplan"
xmlns:ns11="http://domainmodel.pts fsm.org/2015/10/29/fulfilment"
xmlns:ns10="http://domainmodel.pts fsm.org/2015/10/29/transportation"
xmlns:ns16="http://servicemodel.pts fsm.org/2015/10/29/fsm.common.headers"
xmlns:ns15="http://servicemodel.pts_fsm.org/2015/10/29/rsp.booking.messages"
xmlns:ns14="http://domainmodel.pts_fsm.org/2015/10/29/passenger">
   <ns15:Booking>
      <ns3:BookingCommercialRef>836011050593</ns3:BookingCommercialRef>
      <ns3:CarrierServiceItem>
         <ns3:Product QualityOfServiceId="FIRST_CLASS"/>
         <ns3:ProductOwner AddresseeId="83">
```

```
            <ns7:ContractualCarrier CarrierName="Trenitalia" Acronym="83"/>
         </ns3:ProductOwner>
         <ns3:PlaceReference>
            <ns10:CoachId>092</ns10:CoachId>
         </ns3:PlaceReference>
         <ns3:Fare FareTypeId="RESERVATION_ONLY">
            <ns5:RSPPrice>
               <ns5:Currency>
                  <ns5:CurrencyId>EUR</ns5:CurrencyId>
               </ns5:Currency>
               <ns5:Amount>
                  <ns5:FlatFeeAmount>4861</ns5:FlatFeeAmount>
               </ns5:Amount>
            </ns5:RSPPrice>
         </ns3:Fare>
      </ns3:CarrierServiceItem>
      <ns3:Itinerary>
         <ns12:Segment>
            <ns12:Origin StopPlaceId="8308409" Name="ROMA TERMINI">
               <ns9:ScheduledStopPoint>
                  <ns9:EffectiveDeparture>2017-10-20T08:00:00</ns9:EffectiveDeparture>
               </ns9:ScheduledStopPoint>
            </ns12:Origin>
            <ns12:Destination StopPlaceId="8312002" Name="PALERMO CENTRALE">
               <ns9:ScheduledStopPoint>
                  <ns9:ArrivalTime>2017-10-20T10:00:00</ns9:ArrivalTime>
               </ns9:ScheduledStopPoint>
            </ns12:Destination>
            <ns12:Vehicle VehicleId="1955">
               <ns10:ServiceBrandId>InterCityNotte_Italia</ns10:ServiceBrandId>
            </ns12:Vehicle>
         </ns12:Segment>
      </ns3:Itinerary>
   </ns15:Booking>
</ns16:PreBookResponse>
```


The *BookingCommercialRef* element contains a value that is created by combining two different 918 attributes (*SendingSystem* and *LocalReference*) with the help of a SPARQL query.

The value **1** of the attribute *Code* in the *Place* element (used in 918) is converted to the value **FIRST_CLASS** in the attribute *QualityOfServiceId* in the *Product* element with the help of a SPARQL query and master / look up data in the ontology.

The value **83** in the *Carrier* element (used in 918) is converted to the value **Trenitalia** in the attribute *CarrierName* in the *ContractualCarrier* element with the help of a SPARQL query and master / look up data in the ontology.

The value **ROMA TERMINI** in the *DepartureStationName* element (used in 918) is converted to the value **8308409** in the attribute *StopPlaceId* and to the value **ROMA TERMINI** in the attribute *Name* in the *Origin* element with the help of a SPARQL query and master / look up data in the ontology.

The value **PALERMO CENTRALE** in the *ArrivalStationName* element (used in 918) is converted to the value **8312002** in the attribute *StopPlaceId* and to the value **PALERMO CENTRALE** in the attribute *Name* in the *Destinatio*n element with the help of a SPARQL query and master / look up data in the ontology.

The value **96** in the element *Code* in the *ServiceBrand* element (used in 918) is converted to the value **InterCityNotte_Italia** in the *ServiceBrand* element with the help of a SPARQL query and master / look up data in the ontology.

The default values of some FSM elements or attributes are assigned by a SPARQL query, because there are no equivalent values in the 918 message:

- CurrencyId = EUR
- FareTypeId = RESERVATION_ONLY

5. CONCLUSION

To prove that the demonstration platform is ready, all the following steps have been successfully completed:

1. FSM request is sent to Broker (NIKLAS) through the SoapUI tool,
2. Broker (NIKLAS) forwards FSM request to Converter,
3. Converter converts FSM request to 918 request and returns it to Broker (NIKLAS),
4. Broker (NIKLAS) sends the 918 request to Boomerang,
5. Boomerang creates 918 reply and sends it to Broker (NIKLAS),
6. Broker (NIKLAS) forwards 918 reply to Converter,
7. Converter converts 918 reply to FSM response and sends it back to Broker (NIKLAS),
8. Broker (NIKLAS) sends the FSM response to SoapUI tool.

As shown in the Figure 4 – Sent FSM request, received FSM response through SoapUI toolFigure 4, the SoapUI successfully received the FSM response, i.e. the communication loop was completed.

REFERENCES

[1] SoapUI by SMARTBEAR. Available at: https://www.soapui.org/open-source.html

[2] D5.1 - Requirements for the Demo